The Use of Media Characteristics and User Behavior for the Design of Multimedia Servers[1]

D. Venkatesh and T.D.C. Little

Department of Electrical and Computer Engineering
Boston University, Boston, Massachusetts 02215, USA
(617) 353-9877
{*dinesha,tdcl*}*@bu.edu*

MCL Technical Report No. 12-20-1995

Abstract– In this chapter we examine storage management issues in the context of a multimedia information system supporting thousands of users and stored media objects. The design of an efficient MMIS storage server must consider the effects of user access behavior as well as the bandwidth and storage requirements of the stored media objects on system performance. Our presentation addresses these issues by describing a policy for resource allocation and replication yielding efficient operation of the MMIS server. Resulting observations are qualitatively compared against alternative storage hierarchies for large scale multimedia servers constructed in a centralized or distributed fashion. We subsequently examine models for evaluating user access behavior and their application in server design. The chapter concludes with a discussion of the impact of the World Wide Web on the design of future MMIS server architectures.

[1] *Multimedia Information Storage and Management*, S.M. Chung, ed., Kluwer Academic Publishers, 1996, pp. 95-116. This work is supported in part by EMC Corporation and the National Science Foundation under Grant No. IRI-9502702.

1 Introduction

Much of the initial hype on interactive multimedia systems has focused on systems that provide entertainment services to consumer homes such as video-on-demand (VOD) [12, 20]. However, home entertainment represents only a small fraction of the multitude of existing and potential interactive applications. Interactive multimedia information systems (MMISs) are rapidly evolving to encompass diverse application domains such as home-shopping, distance learning, health care, and digital libraries. Designing an interactive MMIS requires bringing together many technologies. These include high speed networks, storage servers, and set-top boxes; the hardware components as well as the software necessary to manage these hardware components. For widespread deployment and usage, the MMIS must support the information needs of a diverse user population. In addition, a MMIS must support complex media formats including audio and video that require relatively large communication bandwidth and storage space. Significant effort is currently underway to build the hardware and software components necessary to build and operate these systems.

It is desirable for a MMIS to support interactive features such as virtual VCR functions on continuous media data (reverse, forward, pause, and stop) and the ability to seek extra information via hyperlinks. From a user's perspective, the creation and storing of information is of little value in itself. Techniques that allow the user to retrieve information quickly and on-demand enhance the utility of the MMIS to the user. Existing alphanumeric database systems support extensive interactive functionality. However, they are poorly equipped to handle the needs of continuous media types such as audio and video. The MMIS server must incorporate mechanisms that map user choices to sessions from devices that store the selected multimedia objects in an efficient manner.[2] From a storage perspective, a MMIS server has finite resources as measured in available storage capacity and I/O bandwidth that limit its ability to support a diverse customer population. These limitations necessitate the development of efficient techniques to manage available storage hardware for multimedia content/information.

In the following sections we examine storage management techniques in a MMIS storage server supporting a large user population.

[2] A *session* is a period during which the user interacts with the MMIS and retrieves data.

2 Issues in the Design of a MMIS Server

The design of the storage server plays an important role in the success of the MMIS. Some important factors that must be considered when designing the storage architecture include [9, 17]: (i) Storage hierarchies that support distributed and heterogeneous databases and allow users access to a wide variety of information, (ii) Staging, caching, and clustering methods for data management, and (iii) Layered access mechanisms that allow users to quickly retrieve the desired information.

Efficient server operation can be achieved by providing the server with mechanisms to collect access statistics for predicting user behavior and map the observations to a storage hierarchy for organizing the media objects in a manner that satisfies the access demands in the best possible way. Accordingly, the important issues that must be considered when developing a MMIS storage-server architecture are:

- *Capacity and Bandwidth Estimation:* Due to the limited I/O bandwidth and storage capacity of a single storage device, supporting the access demand for a large number of users requires aggregating the bandwidths of multiple storage devices [7]. The design of a MMIS must consider the storage device types and organization in addition to other factors including expected system load, media characteristics (such as data rates and timing requirements) and system cost.

- *Resource Allocation and Replacement:* In addition to the hardware components, appropriate mechanisms for data management are necessary. Resources must be allocated to storage devices for maximizing the availability of information to the user. If an imperfect object-storage or user-storage mapping is used, the limitations in I/O bandwidth prevent perfect system utilization. As a result, it is important to develop mechanisms that can intelligently move data around the system so they are available to the users who need them the most. Additionly, the policies to replace obsolete objects when new ones manifest themselves must also be designed with care.

It is clear from our discussion that the design of a MMIS server must consider two important issues (i) how to design an efficient storage architecture and (ii) given the limitations of a storage system, how to manage its resources. These requirements complicate the design of a MMIS as the two issues are not mutually exclusive. In the following sections we examine the different components that constitute a MMIS in great detail and discuss various options available to a designer for building such a system.

2.1 Bandwidth and Capacity Estimation

A MMIS server can be visualized as an array of storage devices, each containing a finite number of media objects. Let N represent the number of users expected to access the system at any given time, K the number of stored objects, and d the number of storage devices in the system. Parameter K can vary over time but is fixed during an operational interval Υ between server updates. To determine the capacity requirements for a MMIS, we consider the relative popularities of the media objects. We begin by noting that a reasonable approach for partitioning the server is based on the relative popularity of a given media object. We can define a popularity vector P consisting of the normalized access demands for the K objects,

$$P = [p_1, p_2, ..., p_{K-1}, p_K] \quad where \quad \sum_{i=1}^{K} p_i = 1[21].$$

The exact form of P is unpredictable due to the random behavior of user accesses.[3] P is non-stationary and changes over time due to the changing popularities of the media objects. It is intuitively apparent that there is a one-one correspondence between the popularity and I/O requirements of a media object. It makes sense to assign a higher bandwidth capacity to the more popular object through replication or by assignment to a faster device.

Estimating the capacity and bandwidth requirements requires a priori knowledge about the nature of user accesses and the distribution of media units in addition to their bandwidth and storage requirements. When this information is not available, one must guess the nature of user accesses to design the system. For purposes of analysis we assume that the distribution of P and the object attributes are determined beforehand. If there are N customers in the system, the number of requests for a given object i can be approximated by Np_i. Let

$$o_j = (x_j, y_j) \quad j \in K$$

represent the bandwidth and storage requirements of a media object j. The system bandwidth necessary to support this object is then given by

$$B_j = Np_jx_j.$$

[3]However, for a large object set, the long term behavior of user accesses can be predicted using the Zipf distribution [19].

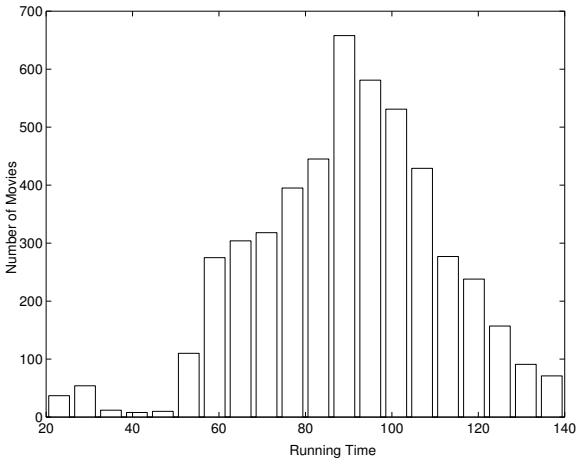

Figure 1: A histogram of movie running times obtained from the *Internet Movie Database* pruned to include only movies released in the US that are longer than 20 mins and shorter than 150 mins.

The cumulative bandwidth requirements that the system must support can be then estimated as

$$BW = \sum_{j=1}^{K} B_j.$$

Developing a similar constraint for estimating the capacity requirement is more difficult. Because each storage device has a fixed I/O bandwidth and storage capacity, a given object may need to be replicated or striped across several devices to meet its projected access demand. In the former case, the capacity requirements for object i equals $n_{oi}y_i$, where n_{oi} is the number of object replicas. In the striped case the storage requirement is only y_i. Thus, there is a tradeoff in the capacity requirement depending upon the choice of the server architecture.

For example, Fig. 1 illustrates a histogram of movie running times for 5,000 movies released in the US [16]. The capacity requirement for this database is 5.3 TB or about 1.05 GB per movie, if we assume that the videos are compressed using the MPEG-I compression standard and that only one copy of each movie is available. Similarly, if we assume that 100,000 users access the system, the aggregate bandwidth requirement is 150 Gb/s. However, as shown in the next section, it may not be possible to simultaneously satisfy the bandwidth and capacity requirements due to storage device limitations. Additional requirements imposed by each user and media object can further complicate the design of the system. However,

assumption of homogeneous object sizes and static user facilitates analysis leading to acceptable designs.

2.2 Server Architectures

Storage technologies that can be used for building a MMIS include (i) solid-state memory devices that have a large I/O capacity but are expensive (i.e., RAM), (ii) rotating media such as CD-ROMs and magnetic disks that are relatively inexpensive and have a good I/O performance, and (iii) tape storage which is inexpensive but has poor I/O performance. These systems are called primary, secondary, and tertiary storage systems, respectively. Many existing MMIS servers use rotating magnetic disks as a basis for their design due to their relatively high bandwidth, moderate cost, and good performance. This choice is most applicable for centralized servers where all information is stored and distributed from a single point.

2.2.1 Replicated Storage Servers

We model the MMIS server as an array of d devices, with a capacity for a finite number of media objects. Due to limited I/O bandwidth, each storage device can only support a finite number of concurrent user sessions. Similarly, the device has a finite storage capacity that limits the number of media objects that can be held simultaneously. When device I/O bandwidth constraints are considered (i.e., limited sessions per device), we can satisfy popularity demands by object replication across the set of devices comprising the server.

To simplify analysis, we assume that the server consists of identical disks and media objects. Let b represent the sustained I/O bandwidth and s the storage capacity for each disk in the system. If N users access the server the number of user requests for the playout of an object i is given by $N \times p_i$. We can subsequently estimate the number of disks required to support the ith movie as

$$d_i = \left\lceil \frac{N \times p_i}{l} \right\rceil,$$

where l is the number of concurrent sessions of object i that a storage device can support.[4] The total number of disks required by the server to support the projected access demand is

[4] l can be estimated as $\lfloor \frac{b}{x} \rfloor$ where x is the per session bandwidth requirement.

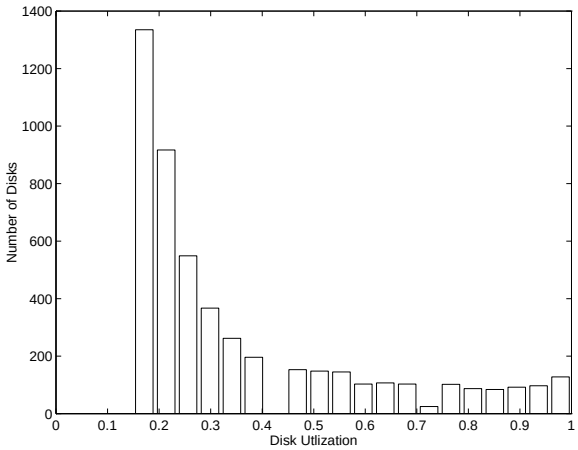

Figure 2: A histogram of disk utilization for user accesses skewed according to the Zipf distribution.

then given by $\sum_{i=1}^{K} d_i$ if each disk can hold a single copy of a media object.

Our analysis assumes that only a single copy of a movie is placed on each disk. This introduces a waste of bandwidth and storage capacity except when all the movies are uniformly popular [6]. For example, if we apply this analysis to a system in which the user accesses are skewed according to the Zipf distribution with $l = 20$, $K = 5,000$, and $N = 100,000$, we can determine that a server employing this design policy requires $8,549$ disks. The resulting distribution of disk utilization is illustrated as a histogram in Fig. 2. We see that a small number of the disks are highly utilized, representing the most popular movies whereas a significant number of the disks are poorly utilized. The average disk utilization is 60%, or 40% of the bandwidth is unused.

An alternate allocation policy is to place multiple objects on each disk. However, as more objects are placed on a storage device, the available I/O for a given object decreases, thereby reducing the chances of a successful session being allocated to the user. This scenario in which a user requests a session and is denied to to the non-availability of resources is termed "blocking." Thus, there is a tradeoff between storage/bandwidth capacity and user satisfaction. It is possible to make some restrictions on object placement that yield bounds on the storage requirements. In other words, there is a tradeoff between the storage capacity of large devices and the I/O capacity of multiple smaller devices. Let us now consider this scenario.

If a given object j must be replicated N_{cj} times, the probability of accessing a given object copy is given by

$$p_{cj} = \frac{p_j}{N_{cj}}.$$

This condition implies that each object-copy is equiprobably accessible. In reality, the actual usage of a object-copy is dependent on dynamic load patterns and server scheduling policies. However, if users are randomly allocated to the object copies we can expect the loading on each copy to be approximately the same.

It can also be shown that there is no reduction in the blocking probability when more than one copy of the same movie is placed on a disk [21]. Separation in user access times yields independent disk head movement as would occur for replicated copies on the same device. Therefore, there is no gain by placing more than one copy of the same object on the same disk. There could be marginal improvement due to reduced latency, but this is not achievable due to the unpredictability of accesses from user to user. A priori knowledge of user interaction would be necessary during physical disk layout to achieve these gains.

Based on the aforementioned discussion, we can describe the storage organization using a popularity-replication matrix Z as a $K \times d$ element matrix such that

$$\sum_{i=1}^{K} z_{i,j} = s_j$$

$$\sum_{j=1}^{d} z_{i,j} = p_i$$

and

$$\sum_{i=1}^{K} \sum_{j=1}^{d} z_{i,j} = 1.$$

Here, s_j is the popularity of disk j being accessed by any user. Row i of Z corresponds to the product of R_i with p_{ic}. The probability of a user's request being successfully satisfied from a disk, Q, is defined as

$$Q = P[\text{ less than } l \text{ sessions on the disk}].$$

To ensure that a maximum number of sessions are supported, Q must be maximized. The probability of a customer's request resulting in a successful connection can be maximized by distributing the media objects among the disks such that each disk has a uniform probability of being accessed [21]. In reality, the problem of allocating movies to disks to balance probabilities is analogous to the "bin-packing" problem that is known to be NP hard. However, as Figs. 3 and 4 illustrate, a moderately balanced system still yields significant performance gains as measured by the number of sessions supported.

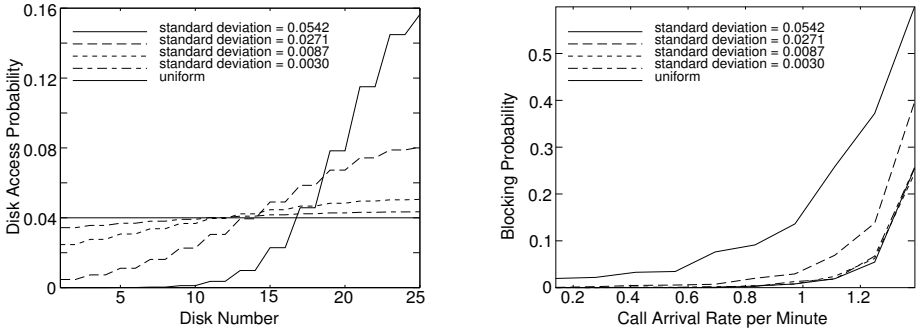

Figure 3: Effects of Load Imbalance in a 25 Disk System for different popularity assignments.

The sensitivity of a server to imbalanced object-disk assignments is shown in Figs. 3 and 4 [21]. The figures illustrate the probability of a user's request being rejected for different disk popularity orderings under two scenarios. In the first case, we consider a system with only 25 disks. In the second case, we study the behavior for a 100 disk system. In both cases, the system model represents a system in which each disk is capable of supporting a maximum of 5 concurrent sessions with an average viewing time of 90 minutes.

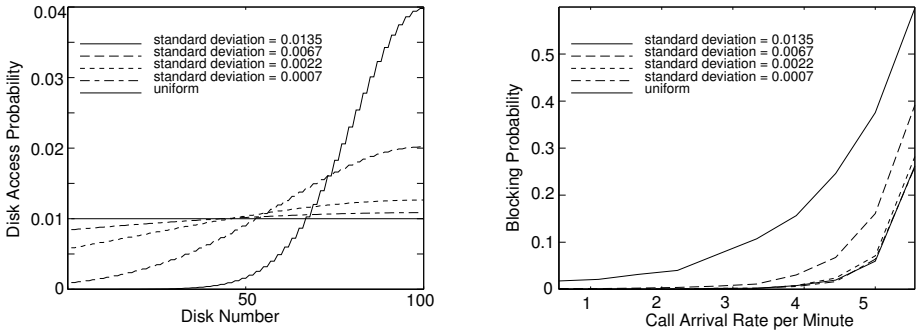

Figure 4: Effects of Load Imbalance in a 100 Disk System for different popularity assignments.

It is apparent from the figures that the balanced system always yields the best performance. An equiprobable assignment of objects to disks always yields the highest session availability.

Furthermore, even moderately "flat" assignments within a reasonable tolerance (10–20 %) yield significant gains and are practically indistinguishable from the optimal.

The advantage of the replicated disk system is that it can support a diverse user population. As objects are replicated across the system to satisfy access demands, it is easy to support different mixes of media objects as well as user access patterns. It is also inherently more reliable as the failure of a single disk affects only the sessions being served off the failed disk. Furthermore, as each user session is scheduled from a single disk, it is more straightforward to provide users with full interactive features including VCR functionalities. The replicated system is also more flexible as additional bandwidth requirements can be satisfied by adding more disks to the system. Disks operate independently and storage reorganization can be done without affecting existing user sessions.

The main disadvantage of the replicated system is its cost. It has been demonstrated that the replicated system is cost effective only under conditions when the distribution of access patterns matches exactly with the placement of media copies [8]. However, finding an ideal mapping for a given access distribution and a set of movies is a difficult problem. In the following sections, we consider alternate architectures and qualitatively evaluate their performance.

2.2.2 Striped Storage Servers

In a striped storage server, a media object is broken into segments and distributed among several storage devices. A scheduler ensures that the data are delivered to the user continuously by periodically switching the session to the correct disk. This is achieved by specifying scheduling intervals during which the system retrieves data for a session and places them in a buffer. As long as the scheduler retrieves data before the buffer can empty, session continuity is ensured [3].

Striped servers are typically more bandwidth constrained than storage constrained. As the data is striped across all the disks in the system the entire system I/O bandwidth can be effectively utilized. This distribution yields a system that is inherently balanced. Performing an analysis similar to the one used in the Section 2.2.1, we determine the number of disks necessary to support the desired storage and bandwidth capacity as

$$d = max\left\{\frac{BW}{b}, \frac{DS}{s}\right\},$$

where s is the capacity of a single disk. A server is storage or bandwidth constrained,

depending on which of the factors, $\frac{BW}{l}$ or $\frac{DS}{s}$ is greater.

The main advantage of a striped system is that the entire system bandwidth can be efficiently utilized, irrespective of the relative popularities of the different objects. The main drawback is the requirement for a sophisticated admission control and scheduling algorithms to ensure that the disks are evenly loaded. This results in a long session startup latency which translates to a long customers waiting time. A second factor is the relatively large buffering requirement necessary for most striped systems which increases their operational cost. The choice of a data layout policy and the enforcement of a scheduling mechanism can decrease this requirement considerably, as has been demonstrated with staggered striping [4]. Most architectures for video servers use a striped architecture for the delivery of video [4, 5, 18, 27].

Striped storage systems are on the average more economical than replicated systems, i.e., their cost per bandwidth ratio is less in comparison with replicated systems. However, they are also less reliable and more inflexible. The failure of a single disk in the system leads to a degradation of service for the entire user population. The use of redundancy reduces the penalty of failure at an increased cost to the end user. It is also difficult to provide interactive features without significantly increasing the processing overhead. A related issue that must be noted is that striped systems typically make use of storage devices with identical size and performance characteristics. It is difficult for heterogeneous devices to be combined in a striped system. Lastly, striped systems are more difficult to scale. Addition of extra devices to increase I/O capacity necessitates a reorganization of all objects stored on the existing disks.

2.2.3 Hierarchical Storage Servers

Replicated and striped disk arrays represent systems in which all data are stored and delivered from a single access point. An alternative approach is to satisfy the capacity and I/O bandwidth constraints by including a diverse set of storage devices and partitioning the storage into a hierarchy [11, 22, 23]. The partitioning of the storage components depends on the distribution of object popularities as well as the cost of the individual storage components.

An example of a hierarchical system is shown in Fig. 5 with several repositories of information interconnected by networks. A hierarchical server can utilize observations of temporal and regional behavior to develop a policy for memory management as is prevalent

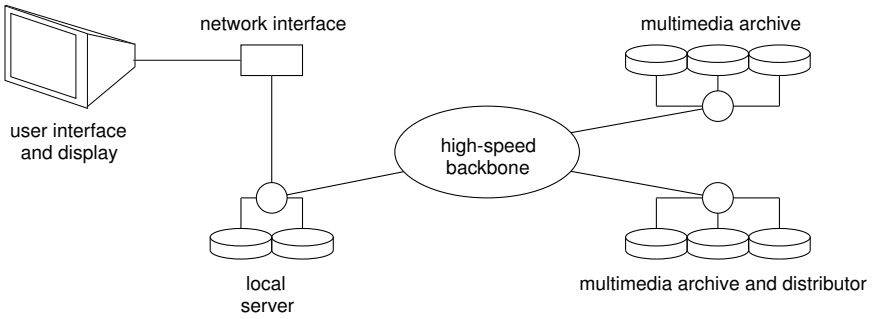

Figure 5: A Hierarchical VOD Architecture

in computer systems [15]. In a computer, locality in program behavior is used to improve performance by storing the information most likely to be utilized in high speed storage for fast access. A similar policy can be developed for the MMIS storage hierarchy to replicate and place information where it is most likely to be utilized. However, managing the storage hierarchy must also consider the real-time nature of information transfers [17]. Some of the advantages of the hierarchical organization, especially in the context of a information system dispersed across a large area and accessed by several thousands of users include:

- The distributed implementation increases the availability of information to the user by supporting several access points. Such a system is also inherently more reliable [20].
- Each local subsystem can be tuned independently for user tastes within the geographic domain and can yield better overall system performance [13].
- Management becomes easier as each system handles its own billing and accounting [20].
- The system can be constructed in a regional-piecewise fashion leading to easy scalability [20].

An additional benefit of a hierarchical scheme is that it takes advantage of the network connectivity bandwidth. A main drawback of a hierarchical architecture is that users get different grades of service depending on the location of the media object within the storage hierarchy. Users accessing the most popular objects stored locally will receive the best quality of service while those accessing the remote devices will incur longer waiting periods and fewer interaction options. A second drawback is that the hierarchical scheme results in good utilization only when the user access demands exactly match the storage hierarchy. A hierarchical scheme is also difficult to evaluate from a reliability standpoint.

2.2.4 Scalable Servers

Instead of providing a dedicated connection to each user, an alternative approach to building a MMIS server is reduce the number of interactive capabilities offered to the user. This can be achieved by not fully supporting interactive features or by scaling down sessions gracefully during periods of overload [6, 24]. Approaches for building scalable servers include batching techniques and service scaling down sessions during periods of overload:

Delayed Batching

In delayed batching, user sessions are delayed for a finite duration so that requests for a given media object can be accumulated and then satisfied en masse [10]. The desired information is subsequently simulcast to all users reducing the I/O requirements. The batching delay depends on several factors including user behavior such as reneging and available system bandwidth. Interactivity can be provided via *contingency channels* that are reserved to handle interactivity. However, the availability of such a channel cannot be guaranteed to every user in the system [25].

Near Video-on-Demand

In Near Video-on-Demand (N-VOD), time is slotted and users receive movies only at the beginning of a slot. Users are allowed to perform interactive VCR functions by switching to contingency channels. Upon the completion of an interaction, the system attempts to merge the user to a N-VOD channel that is the closest in time. Smooth merging of the interactive stream is achieved via buffering. Division of the time into slots allows the system to bound the merges, thereby simplifying its operation [1, 2]. However, just as in the delayed batching scheme, there are no guarantees that all users who desire an interactive channel will receive one.

Adaptive Piggybacking

In adaptive piggybacking, the playout rates of the media streams are varied by speedup or slowdown so that they can be merged with other sessions that are close in time, thus conserving on the I/O bandwidth [14]. The rate of merging must be limited to within $5-10\%$ of the original bandwidth for the effect on the user to be minimal. This requirement places a lower bound on the time difference between two streams that can be merged. An alternate approach is look-ahead scheduling that provides users with pause and resume functionality

[28]. This is achieved by reserving sufficient channels for each media stream and providing a buffer to absorb the delays introduced by interaction.

Statistical Servers
Statistical servers exploit the gains offered by merging the delivery of several variable-bit-rate (VBR) streams. By aggregating the delivery requirements of several streams from the server, it is possible to conserve I/O bandwidth that would be wasted if resources had to be reserved separately for each individual stream [27]. The disadvantage with this approach is that on occasion the combined delivery requirements for all the streams will exceed the I/O capacity of the storage device, resulting in data being dropped.

The main disadvantage of scalable servers is that they perform poorly whenever several users initiate an interaction simultaneously. In addition, managing batching in a large scale commercial server requires significant processing overheads.

By understanding user behavior one can design the storage system and choose the appropriate architecture that can perform satisfactorily. In the next section we address models of user behavior and discuss their utility for the design of a MMIS server.

3 Characterizing User Access Behavior

The design of the storage server for multimedia data depends upon the application under consideration. In most systems including digital libraries and home entertainment, much of the information transfer is unidirectional as the server is primarily used to retrieve data. Bi-directional transfer and storage schemes are necessary in multi-user collaborative environments such as CSCW (computer supported collaborative work) and video editing. The design of the storage hierarchy is affected by the choice of the application domain; systems requiring bidirectional transfer use different design strategies from the unidirectional read-only systems.

The performance of a MMIS server is affected by the dynamics of user accesses behavior. Using static distributions to characterize user access behavior can lead to poor server performance if the accesses are different from that predicted as was demonstrated in Section 2. Using existing models such as those from the video rental stores is a good starting point. However, they do not accurately characterize the temporal, interactive nature of access. For example, the Zipf distribution is most commonly used to characterize the skew in access demands

among the set of available rental videos [1, 10]. However, the Zipf distribution more accurately characterizes the long term behavior of access demands [19]. There will be short term variations that affect the performance of the system and must be accounted for by the designer. This is because the demand for a movie can fluctuate with time due to releases of new movies and other real-world events. Furthermore, these models are not truly representative of the interactive nature of access one would expect in a MMIS server. They are often based on access statistics from the video store or a library where there are a limited number of titles and the user is often limited by the non-availability of a physical copy. Therefore, the data collected based on these studies is of limited use for modeling a system with total interaction and unlimited object availability.

3.1 An Example of a User Model for the WWW

It can be seen that better models for characterizing user behavior are necessary to aid in the design of a MMIS server. In the recent past, the World Wide Web (WWW) has quickly evolved as the most popular means for interactive information dissemination on the Internet.

We see the current access paradigm on the WWW in which the user has complete control over the session as a precursor to future interactive video delivery systems. As the WWW is currently bandwidth constrained rather than server constrained it becomes an ideal candidate to study access demands at a server. Building a more realistic model for different application domains would require tracking the nature of requests to WWW servers that support different data formats and correlating the observed behavior to our system model.

For example, accesses to a WWW server can be used to study the relative loading of the server at various times in a day. Fig. 6 demonstrates the access demand on the server at the Multimedia Communications Laboratory at Boston University for a 24 hour period averaged for a year. Here, the data are not normalized to consider the differences in the time-zones of users accessing the system (although the majority of the accesses were from North America). It is apparent that the loading of the Web server is not uniform. It is at a minimum during the early morning hours, gradually increases during the day and peaks during the late afternoon, after which it falls off. For a home entertainment system, one would expect the accesses to follow a similar pattern, but skewed to the right representing the hours when customers are at home.

The daily access behavior can be used to schedule operational periods and develop a pricing policy for a MMIS server. The number of requests to the video server will be least

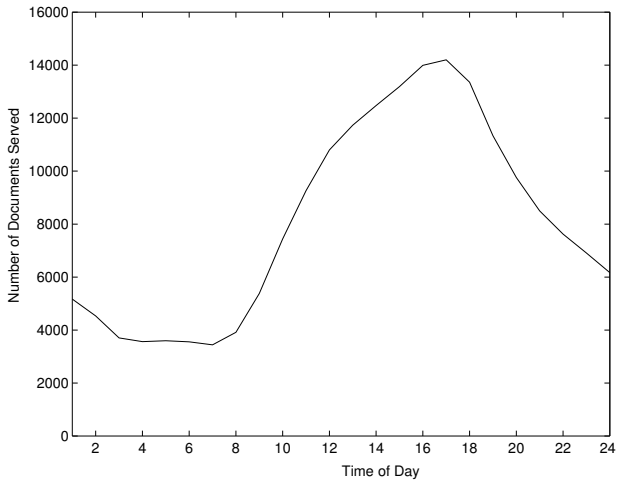

Figure 6: Server Loading vs. Time of Day for the MCL WWW Server

during the early morning hours. This time can be scheduled to estimate future loads and reorganize the video server to satisfy future demands. A pricing policy can be used to attract customers during the slack periods by offering them lower costs. Such a policy increases revenue and reduces server congestion. Studies on user reneging behavior can be used in developing batching approaches to aggregate sessions and conserve server I/O bandwidth during peak loads [26].

3.1.1 Object Popularity

A second factor that affects the performance of the MMIS is the variation of an object's relative popularity with time. Typically, a object's relative popularity at a given instant demonstrates a gradual decline with time. However, this behavior is not smooth and can change drastically because of a published review or a publicity blitz. A MMIS server must absorb such load fluctuations with little visible effect on the end user. For example, on a server employing a caching technique, the implications of a sudden load surge imply a phase of operation during which the session blocking probability is increased. In a striped system, the admission control algorithm must modify its behavior to ensure that the disks are evenly loaded. A load fluctuation also necessitates a redistribution of video copies in a replicated system.

The computation of an object's relative popularity can be done on a periodic basis. The frequency of computation depends upon the rate at which the popularity changes. For example, a daily computation of the popularities is sufficient to track the dynamics of a

movie database while it may have to be tracked hourly for a news database. A predictor is necessary to map the observed popularities to future database accesses. This can be achieved by a simple linear predictor. If $N_i(t-2)$ and $N_i(t-1)$ represent the number of requests for object i for periods $(t-2)$ and $(t-1)$ respectively, the number of requests that will occur in period t can be estimated as:

$$N_i(t) = 2N_i(t-1) - N_i(t-2), \quad N_i(t) \geq 0. \tag{1}$$

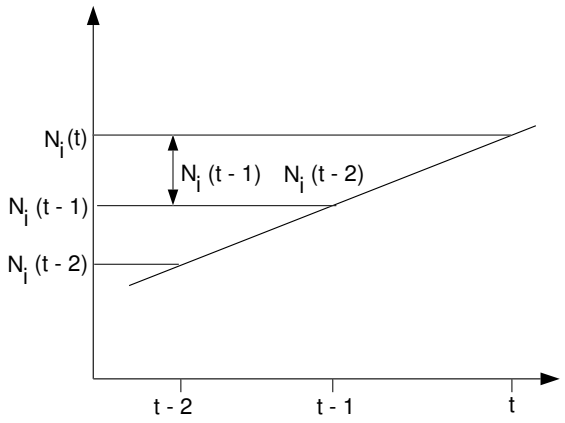

Figure 7: A Linear Predictor

In other words, $N_i(t)$ is a linear function of the number of requests for i in the previous two periods. The slope is determined by the change in the number of users. An example of the behavior of a linear predictor is illustrated in Fig 7. This simple predictor can be replaced by more complex nonlinear predictors that considers detailed models of user behavior for a more accurate approximation of user behavior and popularity estimation. When new objects manifest themselves, they must be given a predetermined value N_{new} based on prior estimates.

4 Summary

In this chapter we considered issues relevant to the management of large repositories of multimedia information with respect to on-line access models and video-on-demand. We

examined issues of storage management in the context of a MMIS supporting thousands of users and media objects.

In our discussion, we developed a framework for estimating the storage and bandwidth requirements for a multimedia server. We subsequently examined techniques for mapping the projected access demands to available server capacity in an efficient manner. The policies developed in this chapter focused on replicated storage servers. We also examined in detail issues that must be considered in scaling these results to alternate storage architectures including striped, hierarchical, and scalable servers. We can conclude from our study that the storage requirements are application dependent and must be carefully chosen to find an ideal architecture.

We also discussed issues in the design of a MMIS server that can benefit from observations made from the WWW domain. It is our belief that the WWW represents an information rich environment whose behavior can be used to design efficient multimedia servers. There are several interesting issues that require further analysis and can be identified for future research. These include pricing policies and their behavior on server performance, media mapping, and synchronization requirements for complex multimedia presentations.

References

[1] K. Almeroth and M. Ammar, "On the Performance of a Multicast Delivery Video-On-Demand Service with Discontinuous VCR Actions," *Proc. Intl. Conf. on Communications (ICC '95)*, Seattle WA, June 1995, pp. 1631-1635.

[2] K. Almeroth and M. Ammar, "A Scalable, Interactive Video-On-Demand Service Using Multicast Communication," *Proc. Intl. Conf. on Computer Communication and Networks (IC3N '94)* , San Francisco CA, September 1995, pp. 292-296.

[3] D.P. Anderson and G. Homsy, "A Continuous Media I/O Server and its Synchronization Mechanism," *Computer*, Vol. 24, No. 10, October 1991, pp. 51-57.

[4] S. Berson, S. Ghandeharizadeh, R. Muntz, and X. Ju, "Staggered Striping in Multimedia Information Systems," *Proc. ACM SIGMOD*, 1994, pp. 79-89.

[5] E. Chang and A. Zakhor, "Scalable Video Data Placement on Parallel Disk Arrays," *Storage and Retrieval for Image and Video Databases II, IS& T/SPIE Symposium on Elec. Imaging Sci. and Tech.*, SPIE Vol. 2185, February 1994, San Jose CA, pp. 208-221.

[6] H.J. Chen, T.D.C. Little, and D. Venkatesh, "A Storage and Retrieval Technique for Scalable Delivery of MPEG-Encoded Video," *Journal of Parallel and Distributed Computing*, Vol. 30., No. 2, November 1995, pp. 180-189.

[7] P.M. Chen, E.K. Lee, G.A. Gibson, R.H. Katz, and D.A. Patterson, "RAID: High-Performance, Reliable Secondary Storage," *Technical Report TR-CS-93-778*, University of California, Berkeley, 1993.

[8] A.L. Chervenak, D.A. Patterson, and R.H. Katz, "Choosing the Best Storage System for Video Service," *Proc. 3rd ACM Intl. Conf. on Multimedia (ACM Multimedia'95)*, San Francisco CA, November 1995, pp. 109-119.

[9] R.A. Coyne and H. Hulen, "National Storage Laboratory: A Collaborative Research Project," *SPIE: Enabling Technologies for High Bandwidth Applications*, 1992, Vol. 1785, pp. 13-26.

[10] A. Dan, D. Sitaram, and P. Shahabuddin, "Scheduling Policies for an On-Demand Video Server with Batching," *Proc. 2nd ACM Intl. Conf. on Multimedia (ACM Multimedia'94)*, San Francisco, October 1994, pp. 15-23.

[11] Y.N. Doganata and A.N. Tantawi, "A Video Server Cost/Performance Estimator Tool," *Journal of Multimedia Tools and Applications,* Vol. 1, No. 2, pp. 185-202.

[12] A.D. Gelman, H.Kobrinski, L.S. Smoot, S.B.Weinstein, M. Fortier, and D. Lemay, "A store-and-forward architecture for video-on-demand service," In *Proceedings of the IEEE ICC*, pages 27.3.1 - 27.3.5, 1991.

[13] A. Gersht and S. Kheradpir, "Real-Time Decentralized Traffic Management Using a Parallel Algorithm," *Proc. IEEE Global Telecommunications Conference and Exhibition (Globecom '90)*, December 1990, pp. 0408-0414.

[14] L. Golubchik, J.C.S. Lui, and R. Muntz, "Reducing I/O Demand in Video-on-Demand Storage Servers," *UCLA Computer Science Tech-report 940037*, 1994.

[15] J.L. Hennessy and D.A. Patterson, "Computer Architecture A Quantitative Approach," *Morgan Kaufmann Publishers Inc.*, Palo Alto CA, 1990.

[16] M. Harding, ed., "Movie Running-Times List," *Internet Movie Database*, August 6, 1995.

[17] R. Jain, ed., "NSF Workshop on Visual Information Management Systems," *ACM SIGMOD RECORD*, Vol. 22, No. 3, September 1993, pp. 57-75.

[18] K. Keeton and R.H. Katz, "The Evaluation of Video Layout Strategies on a High-Bandwidth File Server," *Proceedings of the 4th International Workshop Network and Operating Systems Support for Digital Audio and Video*, Lancaster, England, November 1993, pp. 237-248.

[19] F.W. Lancaster, "The Measurement and Evaluation of Library Services," *Information Resources Press*, Washington D.C., 1977.

[20] T.D.C. Little and D. Venkatesh, "Prospects for Interactive Video-on-Demand," *IEEE Multimedia*, Vol. 1, No. 3, Fall 1994, pp. 14-24.

[21] T.D.C. Little and D. Venkatesh, "Probabilistic Assignment of Movies to Storage Devices in a Video-on-Demand System," *Multimedia Systems*, Vol. 2, No. 6, January 1995, pp. 280-287.

[22] J.-P. Nussbaumer, B.V. Patel, F. Schaffa, and J.P.G. Sterbenz, "Networking Requirements for Interactive Video on Demand," *IEEE Journal on Selected Areas in Communications,* Vol. 13, No. 5, June 1995, pp. 779-787.

[23] R. Ramarao, and V. Ramamoorthy, "Architectural Design of On-Demand Video Delivery Systems: The Spatio-Temporal Storage Allocation Problem," *Proc. ICC'91*, 1991, Denver CO, pp. 17.6.1-17.6.5.

[24] P.J. Shenoy and H.M. Vin, "Efficient Support for Scan Operations in Video Servers," *Proc. 3rd ACM Intl. Conference on Multimedia, (ACM Multimedia'95)*, San Francisco CA, November 1995, pp. 131-140.

[25] D. Venkatesh and T.D.C. Little, "Dynamic Service Aggregation for Efficient Use of Resources in Interactive Video Delivery," *Proc. 5th Intl. Workshop on Network and Operating System Support for Digital Audio and Video* (NOSSDAV'95), Durham, New Hampshire, April 19-22, 1995, pp. 119-122.

[26] D. Venkatesh and T.D.C. Little, "Investigation of Web Server Access as a Basis for Designing Video-on-Demand Systems," *Proc. First Intl. Symposium on Photonics Technologies and Systems for Voice, Video, and Data Communications*, Philadelphia PA, October 1995, SPIE Vol. 2617-06.

[27] H.M. Vin, A. Goyal, A. Goyal and P. Goyal, "An Observation Based Admission Control Algorithm for Multimedia Servers," *Proc. 1st IEEE Intl. Conf. on Multimedia Computing and Systems (ICMCS'94)*, Boston MA, May 1994, pp. 234-243.

[28] P.S. Yu, J.L. Wolf, and H. Shachnai, "Design and Analysis of a Look-Ahead scheduling Scheme to Support Pause-Resume for Video-on-Demand Applications," *Multimedia Systems* Vol. 3, No. 4, 1995, pp. 137-149.